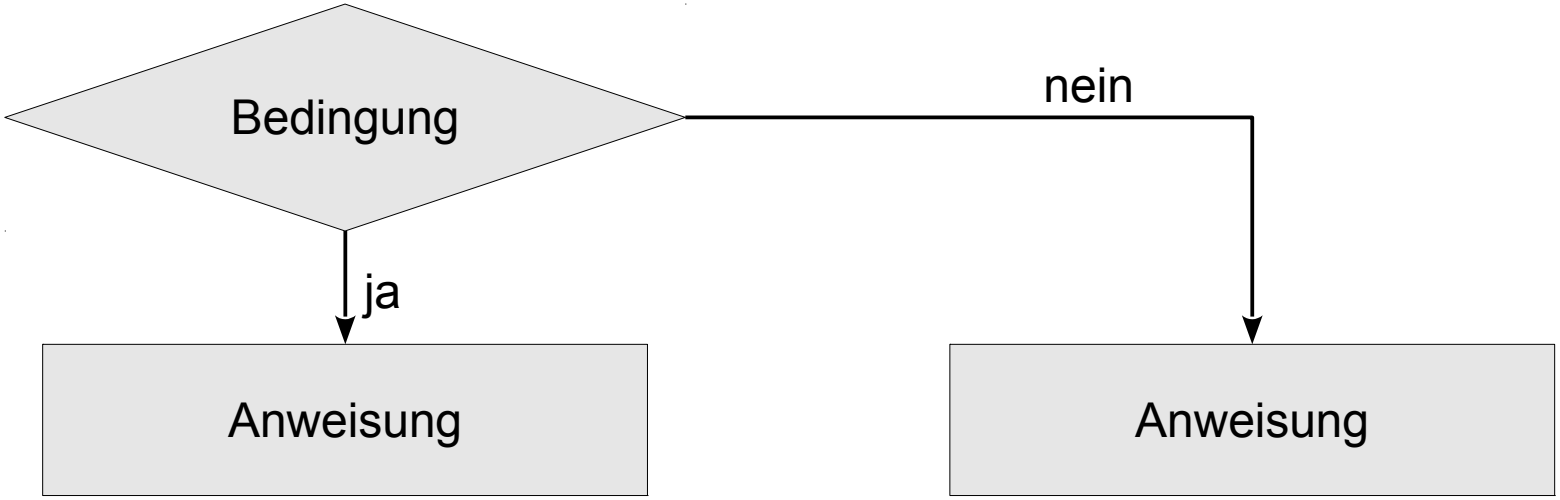
Java - Bedingte Anweisungen



Anweisungen

```
    double dblErgebnis;

    intZahl = 4;
    dblErgebnis = intZahl * DBL_ZAHL;
    System.out.println(dblErgebnis);
```

- Deklaration von Variablen und Konstanten.
- Zuweisungen an Variablen.
- Berechnung von variablen Werten.
- Aufruf von Methoden.

Leerzeichen in Anweisungen

```
    double dblErgebnis;

    intZahl = 4;
    dblErgebnis = intZahl * DBL_ZAHL;
```

- Zwischen Operanden und Schlüsselwörtern muss ein Leerzeichen stehen.
- Zwischen Operanden und Operatoren können Leerzeichen stehen, müssen aber nicht.
- Mit Hilfe von Leerzeichen wird die Lesbarkeit einer Anweisung erhöht.

Anweisungen „beenden“

```
    double dblErgebnis;

    intZahl = 4;
    dblErgebnis = intZahl * DBL_ZAHL;
```

- Jede Anweisung endet mit dem Semikolon.
- Pro Zeile können beliebig viele Anweisungen stehen. Aber es sollte nie mehr als eine Anweisung stehen.

Leere Anweisung

```
    double dblErgebnis;

    intZahl = 4;
    ;
```

- Ein Semikolon kennzeichnet eine leere Anweisung.
- Eine leere Anweisung an den falschen Stellen kann zu Fehlern im Programmablauf führen.

Block von Anweisungen

```
public class Java03_BedingteAnweisung {

    public static void main(String[] args) {
        final double DBL_ZAHL = 0.5;
        int intZahl;
        double dblErgebnis;

        intZahl = 4;
        dblErgebnis = intZahl * DBL_ZAHL;

    }

}
```

Erläuterung

- Beginn und Ende mit den geschweiften Klammern.
- Zusammenfassung von Anweisungen, die in Abhängigkeit einer Bedingung ausgeführt werden.
- Eine Methode fasst Anweisungen zur Beschreibung einer Aktivität zusammen.

Operanden

- Variablen, die einen dynamischen Wert speichern.
- Konstanten, die einen festen Wert speichern. Der Wert einer Konstanten ist nicht durch Anweisungen veränderbar.
- Literale. Statische Werte, die direkt in einer Anweisung stehen. Zahlen wie 4 oder 0.5. Einzelne Zeichen wie ´a´.

Beispiel

```
    public static void main(String[] args) {
       final double dlbZahl = 0.5;   // Konstante
       int intZahl;                  // Variable

       intZahl = 4;                  // Literal 4

    }
```

Operatoren

- Regeln zur Berechnung von Werten.
- Verknüpfung von Ausdrücken und Werten.
- Vorschriften zur Bildung von Ausdrücken aus mehreren Operanden.

Operatoren in Java

- Arithmetische Operatoren berechnen mit Hilfe eines Ausdrucks einen Wert. Der Wert wird einer Variablen zugewiesen, in einer Tabelle gespeichert etc.
- Operatoren vergleichen zwei Werte und geben einen boolschen Wert zurück. Falls der Vergleich stimmt, wird true (wahr) zurückgegeben. Andernfalls wird false (falsch) zurückgegeben. Bedingte Anweisungen und Schleifen in Java nutzen Vergleichsoperatoren.
- Logische Operatoren verknüpfen Ausdrücke, die einen boolschen Wert zurückgeben. Logische Operatoren wie AND, OR und NOT verknüpfen Bedingungen in Anweisungen.

Arithmetische Operatoren

Operator	Berechnung	Beispiel
+	Addition	3 + 4 = 7
-	Subtraktion	3 – 4 = -1
*	Multiplikation	3 * 4 = 12
/	Division	3 / 4 = 0.75
%	Division mit Rest	3 % 4 = 3

Beispiel

```
final double DBL_ZAHL = 0.5;
int intZahl;
double dblErgebnis;

intZahl = 4;
dblErgebnis = intZahl * DBL_ZAHL;
dblErgebnis = intZahl / DBL_ZAHL;
dblErgebnis = intZahl + DBL_ZAHL;
dblErgebnis = intZahl - DBL_ZAHL;
```

Relationale Operatoren (Vergleichsoperatoren)

Operator	Berechnung	Beispiel
==	ist gleich	4 == 5 → falsch
!=	ungleich	4 != 5 → true
<	kleiner als	4 <5 → true
<=	kleiner gleich als	4 <= 5 → true
>	größer als	4 > 5 → false
>=	größer gleich als	4 >= 5 → false

Beispiel

```
int intZahlL = 3;
int intZahlR = 5;
boolean blnErgebnis;

blnErgebnis = (intZahlL == intZahlR);
blnErgebnis = (intZahlL != intZahlR);
blnErgebnis = (intZahlL < intZahlR);
blnErgebnis = (intZahlL <= intZahlR);
blnErgebnis = (intZahlL > intZahlR);
blnErgebnis = (intZahlL >= intZahlR);
```

Nutzung von relationalen Operatoren

- Vergleiche von zwei Operanden.
- Beantwortung von Fragen, auf die mit Ja oder Nein geantwortet werden kann.
- Haben Eigenschaften die Ausprägung? Zum Beispiel: Die hat Katze eine graue Fellfarbe. Die Aussage kann durch Sehen überprüft werden. Falls die Katze eine schwarze Fellfarbe hat, ist die die Aussage falsch.

Hinweise

- Zusammengesetzte Operatoren wie ==, <= und so weiter dürfen nicht durch ein Leerzeichen getrennt werden.
- Der Zuweisungsoperator = darf nicht mit dem Operator „ist gleich“ == verwechselt werden.
- Mit Hilfe von Klammern kann die Lesbarkeit des Ausdrucks erhöht werden.
- Gleitkommazahlen nähren sich einem Wert an. Aus diesen Grund sollte eine Überprüfung auf Gleichheit vermieden werden.

Verknüpfungsoperatoren

Operator	Erläuterung
x || y	x ODER y. Einer der beiden Ausdrücke muss wahr sein. Wenn x true ist, wird y nicht mehr ausgewertet.
x && y	x UND y Beide Ausdrücke müssen wahr sein. Wenn x false ist, wird y nicht mehr ausgewertet.
!x	NICHT x Wahr wird zu falsch und umgekehrt.

Beispiel

```
char cZeichen;
int intZahl;
boolean blnErgebnis;

intZahl = 5;
blnErgebnis = (!(intZahl > 0));
blnErgebnis = ((intZahl >= 0) && (intZahl <= 10));

cZeichen = 'Q';
blnErgebnis = ((cZeichen == 'q') ||
               (cZeichen == 'Q'));
```

Negation

```
int intZahl;
boolean blnErgebnis;

intZahl = 5;
blnErgebnis = (!(intZahl > 0));
blnErgebnis = (intZahl < 0);
```

- Falsch wird zu wahr und umgekehrt.
- Die Negation kann häufig durch eine „Umkehrung“ der Operatoren ersetzt werden. In diesem Beispiel wird der „ist größer“-Operator durch den „ist kleiner“-Operator ersetzt.

UND-Verknüpfung

```
int intZahl;
boolean blnErgebnis;

intZahl = 5;
blnErgebnis = ((intZahl >= 0) && (intZahl <= 10));
```

- Der linke sowohl als auch der rechte Ausdruck müssen wahr sein.
- Sobald einer der Ausdrücke falsch ist, ist auch der Gesamtausdruck falsch.

ODER-Verknüpfung

```
char cZeichen;
boolean blnErgebnis;

cZeichen = 'Q';
blnErgebnis = ((cZeichen == 'q') ||
               (cZeichen == 'Q'));
```

- Einer der beiden Ausdrücke muss wahr sein.

Runde Klammern in Java

```
boolean = (Bedingung)

Variable = (ausdruck) operator (ausdruck)

System.out.println(wert)
```

- Runde Klammern fassen Ausdrücke zusammen.
- Runde Klammern erhöhen die Lesbarkeit von komplexen Ausdrücken.
- Begrenzung der Argumentliste einer Methode wie zum Beispiel `println()`.

Geschweifte Klammern in Java

- Geschweifte Klammern fassen Blöcke von Anweisungen zusammen.
- Die Blöcke von Anweisungen beschreiben eine Aktivität in Einzelschritten.
- Codeblöcke beschreiben ein Objekt vollständig.
- Anweisungsblöcke werden in Abhängigkeit von Bedingungen ausgeführt.

```
{
    Anweisung 1;
    Anweisung 2;
}
```

Prioritäten (Rangfolge)

Priorität	Operator
1	Nicht ! Inkrement ++, Dekrement --
2	Multiplikation * Division / Modulo %
3	Addition + Subtraktion -
4	Kleiner <, Kleiner Gleich <= Größer >, Größer gleich >=
5	ist Gleich ==, ist ungleich !=
6	Und &&
7	Oder ||

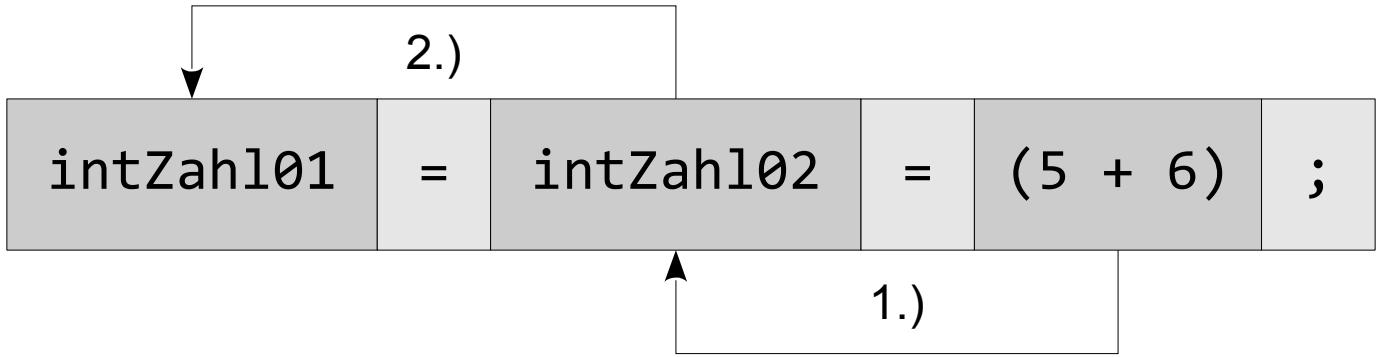
Beispiel

```
double dblZahl = 0.3;
int intZahl = 4;
double ergebnis = 0;

ergebnis = dblZahl * dblZahl + 2 * dblZahl * intZahl +
           intZahl * intZahl;
ergebnis = (dblZahl * dblZahl) + (2 * dblZahl * intZahl) +
           (intZahl * intZahl);

ergebnis = (dblZahl * (dblZahl + 2) * dblZahl * intZahl) +
           (intZahl * intZahl);
```

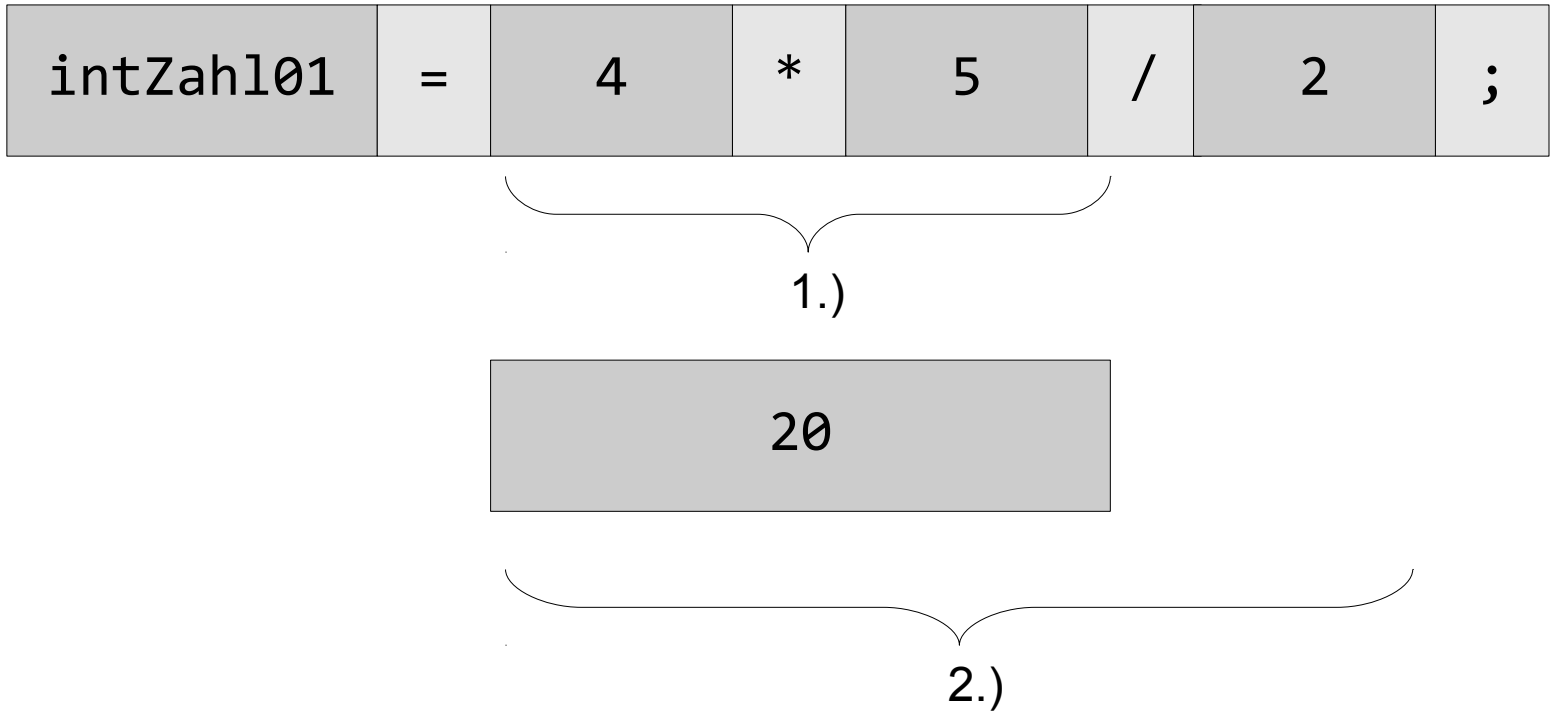
Assoziativität

Assoziativität	Operator
→	Inkrement ++, Dekrement – Multiplikation * Division Modulo % Addition + Subtraktion - Kleiner <, Kleiner Gleich <= Größer >, Größer gleich >= ist gleich ==, ist nicht gleich != Und &&, Oder ||
←	Nicht ! Inkrement ++, Dekrement -- Zuweisung =

Rechtsassoziative Verknüpfung



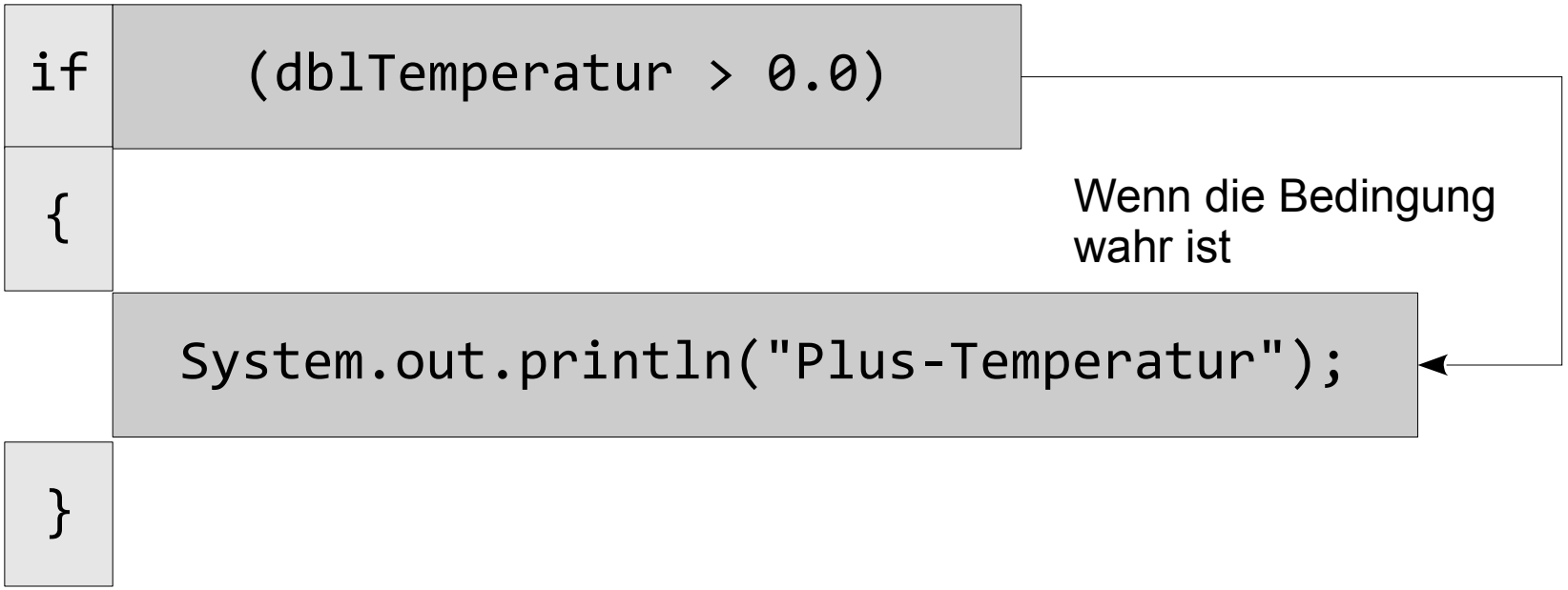
Linksassoziative Verknüpfung



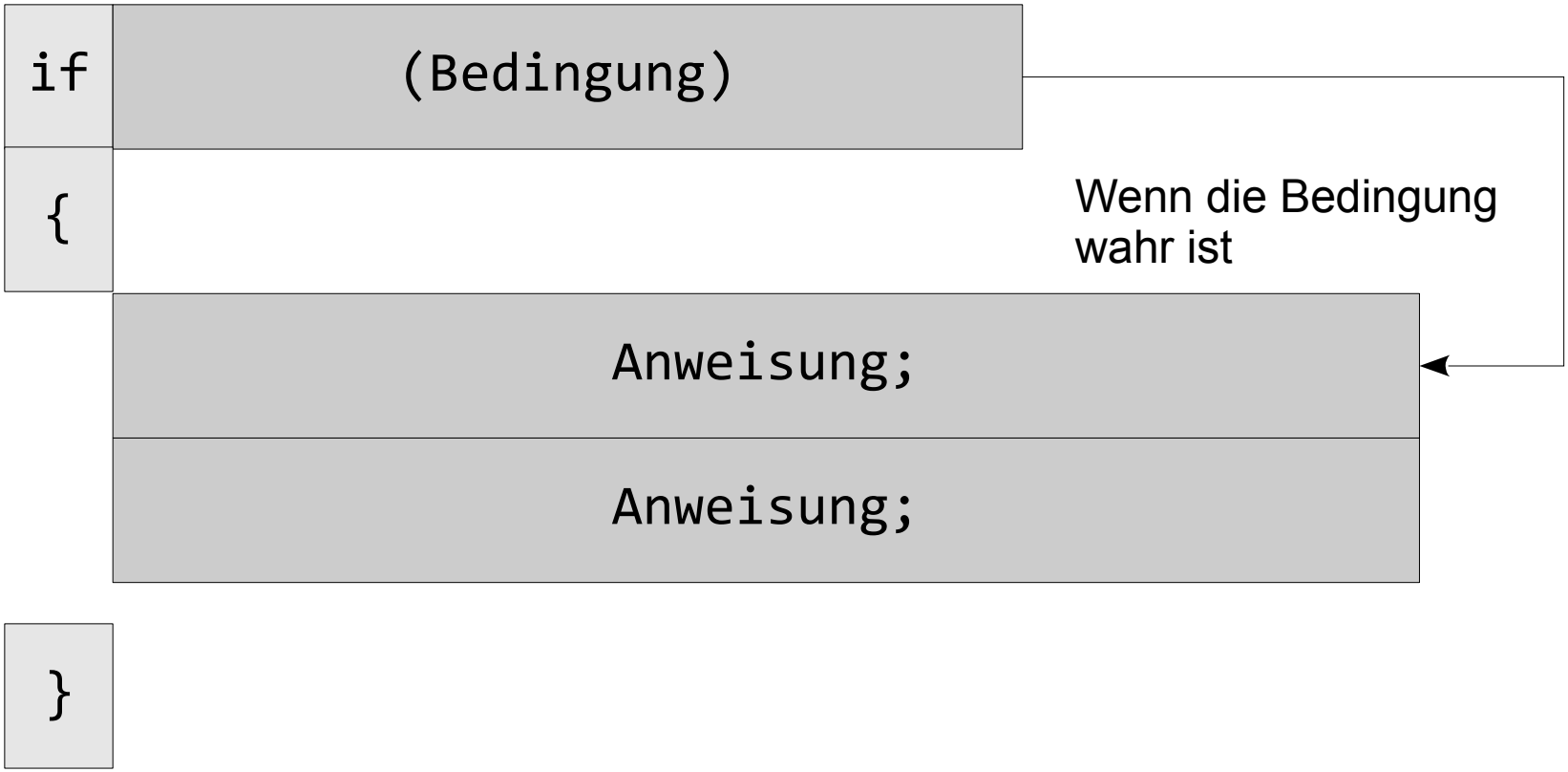
Bedingte Anweisung

```
if (dblTemperatur > 0.0)
{
  System.out.println("Plus-Temperatur");
}
else if (dblTemperatur < 0.0)
{
  System.out.println("Minus-Temperatur");
}
else
{
    System.out.println("Null-Punkt");
}
```

Arbeitsweise



Aufbau


if
(Bedingung)
{
Wenn die Bedingung
wahr ist
Anweisung;
Anweisung;
}

Kopf einer if-Anweisung

`if`	`(dblTemperatur > 0.0)`
`if`	`(Bedingung)`

- Jede Bedingte Anweisung beginnt mit dem Schlüsselwort `if`.
- Wenn die Bedingung zutrifft, führe den dazugehörigen Block von Anweisungen aus.

Bedingungen

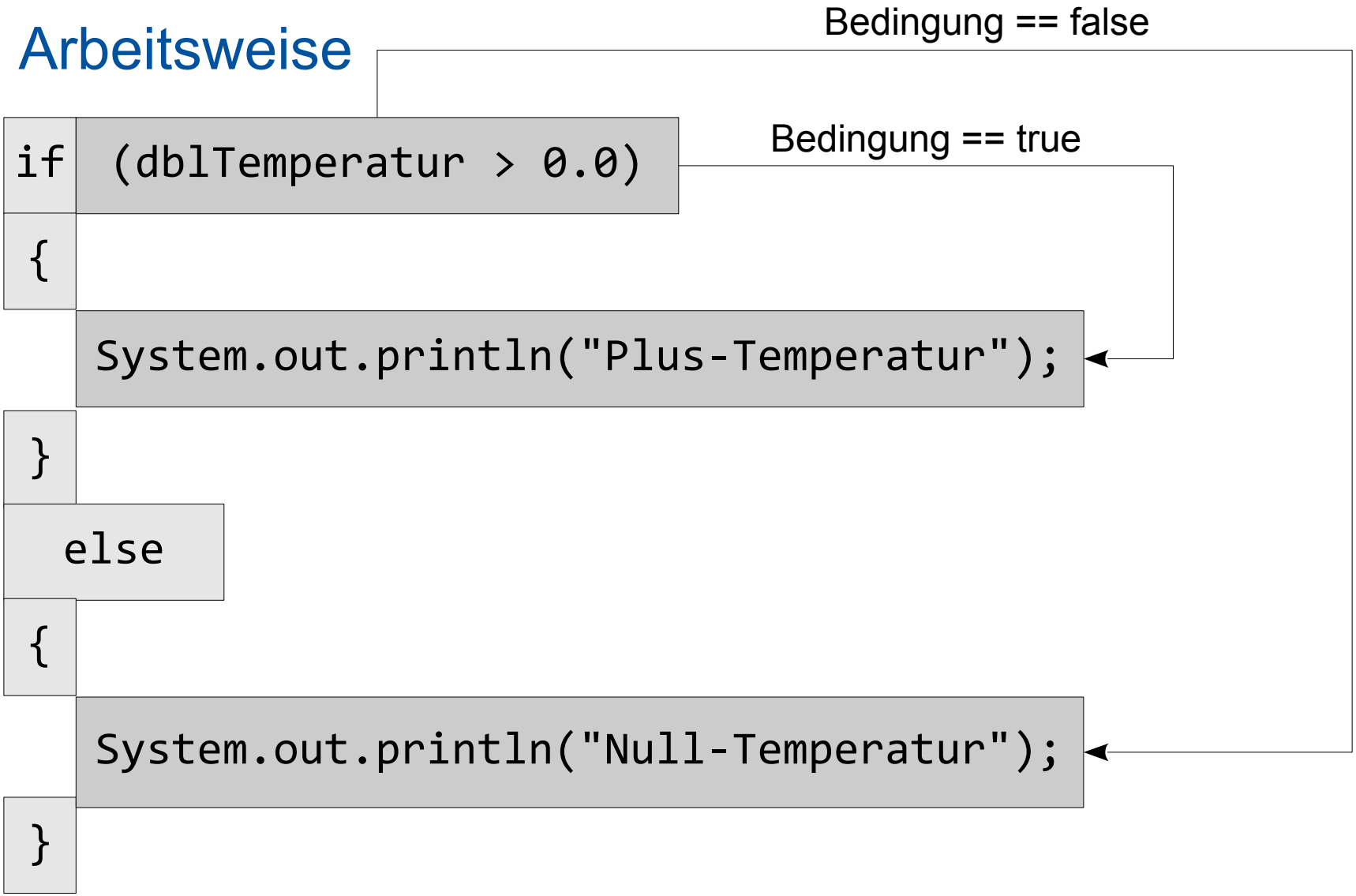
`if`	`(dblTemperatur > 0.0)`
`if`	`(Bedingung)`

- Bedingungen werden aus Operanden und Vergleichsoperatoren gebildet.
- Bedingungen können mit Hilfe von logischen Operatoren verknüpft oder negiert werden.
- Bedingungen müssen geklammert werden.

If – else - Anweisung

```
if (dblTemperatur > 0.0)
{
  System.out.println("Plus-Temperatur");
}

else
{
    System.out.println("Null-Punkt");
}
```

Arbeitsweise



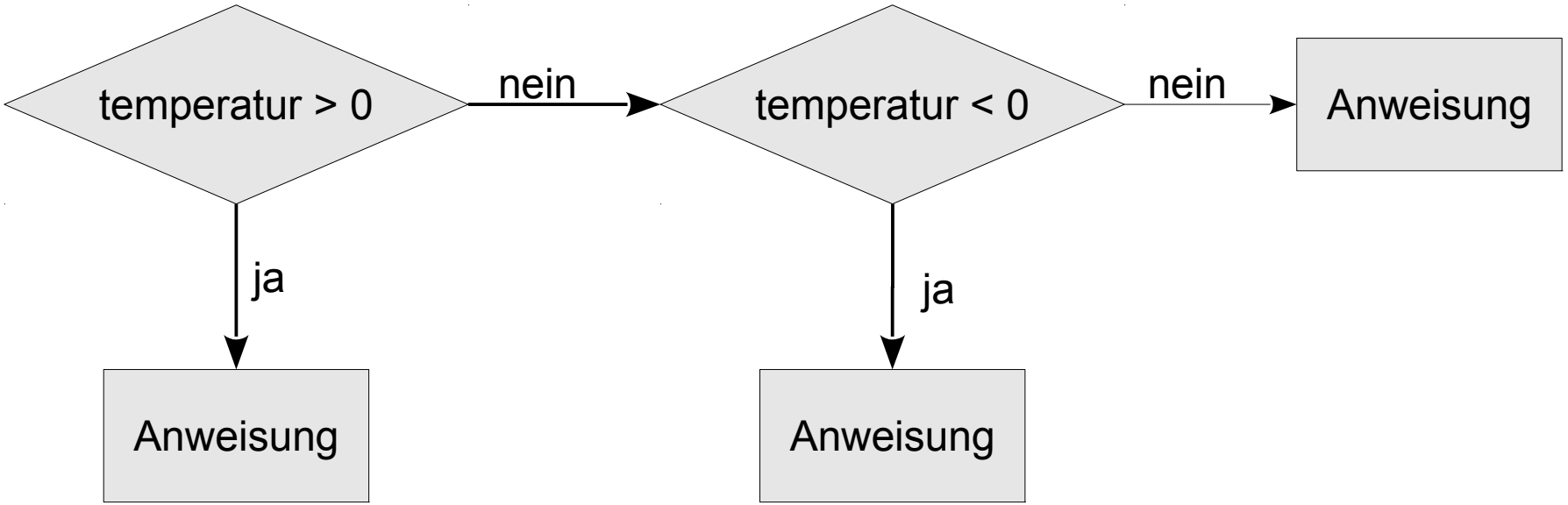
else-Anweisung

- Der Standardfall beginnt mit dem Schlüsselwort **`else`**.
- Wenn die Bedingung zutrifft, dann… Andernfalls…
- In Abhängigkeit der geschweiften Klammern wird die else-Anweisung einer if-Anweisung zugeordnet.

Fallunterscheidung

```
if (dblTemperatur > 0.0)
{
  System.out.println("Plus-Temperatur");
}
else if (dblTemperatur < 0.0)
{
  System.out.println("Minus-Temperatur");
}
else
{
    System.out.println("Null-Punkt");
}
```

Grafische Darstellung



Verknüpfungen von Bedingungen

```
temperatur = 39.456;
if((temperatur >= 36.9) && (temperatur <= 37.4)){
        System.out.println("Erhöhte Temperatur");
}
else if ((temperatur >= 37.5) && (temperatur <= 39.4)){
        System.out.println("Fieber");
}
else if ((temperatur >= 39.5) && (temperatur <= 40.5)){
        System.out.println("Hohes Fieber");
}
else if (temperatur >= 40.5){
      System.out.println("Sehr hohes Fieber");
}
else {
      System.out.println("Normaltemperatur");
}
```

Hinweise

- Die Bedingungen der Fallunterscheidungen müssen auf Korrektheit überprüft werden.
- Bei Gleitkommazahlen müssen die Nachkommastellen bei der Fallunterscheidung berücksichtigt werden.

Schachtelung von bedingten Anweisungen

```
zeichen= 'c';
temperatur = 1.5;

if ((zeichen == 'C') || (zeichen == 'c')){
    System.out.println("Umrechnung in Celsius");

    if (temperatur > 0.0) {
        System.out.println("Plus-Temperatur");
    }
}
```

Hinweise

- Entsprechend der Verschachtelungstiefe können die else if- und else-Anweisungen der if-Anweisung zugeordnet werden.
- Die Einrückungen symbolisiert die Verschachtelungstiefe.

Fallunterscheidung mit switch

```
switch (zeichen)
{
   case 'c':
      System.out.println("Umrechnung in Celsius.");
      break;

   case 'f':
      System.out.println("Umrechnung in Fahrenheit.");
      break;

   default:
      System.out.println("Umrechnung nicht möglich.");
}
```

Kopf einer switch-Anweisung

`switch`	`(zeichen)`
`switch`	`(Variablen-Name)`

- Dem Schlüsselwort `switch` folgt in runden Klammern die zu untersuchende Variable.
- Es können Variablen vom Datentyp `byte, short, int, char` und Instanzen der Klasse `string` seit Java 7 sowie Aufzählungskonstanten untersucht werden.

Zu untersuchende Fälle

```
switch (zeichen)
{
   case 'c':
      System.out.println("Umrechnung in Celsius.");
      break;
}
```

- In dem Block von Anweisungen zu eine switch-Anweisung werden die zu untersuchenden Fälle aufgeführt.
- Jeder Fall beginnt mit dem Schlüsselwort `case`.
- Die Fälle werden von oben nach unten bearbeitet.

Kopf eines Falls

`case`	`'c'`	`:`
`case`	`Wert`	`:`

- Dem Schlüsselwort `case` folgt der Vergleichswert.
- Die Beschreibung des Falls endet mit dem Doppelpunkt.
- Der Vergleichswert kann entsprechend des Datentyps des zu untersuchenden Wertes interpretiert werden.

„Beendigung“ des Falls

```
switch (zeichen)
{
   case 'c':
      System.out.println("Umrechnung in Celsius.");
      break;
}
```

- Der Fall endet mit dem Schlüsselwort `break`.
- Mit Hilfe des des Schlüsselwortes `break` wird die Verarbeitung der Anweisungen in einem Block abgebrochen.

Standardfall (Default)

```
switch (zeichen)
{
   case 'c':
      System.out.println("Umrechnung in Celsius.");
      break;

   default:
      System.out.println("Umrechnung nicht möglich.");
}
```

Hinweise

- Wenn alle beschriebenen Fälle nicht zu treffen, wird der Standardfall ausgeführt.
- Der Standardfall beginnt mit dem Schlüsselwort `default`. Der Standardfall ist das letzte Element in einer Reihenfolge von Fällen.
- Der letzte Fall in einer Reihenfolge von Fällen muss nicht mit dem Schlüsselwort `break` abgeschlossen werden.

Zusammenfassung von Fällen

```
switch (zeichen)
{
   case 'c':
   case 'C':
      System.out.println("Umrechnung in Celsius.");
      break;

   default:
      System.out.println("Umrechnung nicht möglich.");
}
```